**Setting and Unsetting Variables**

Bash has two built-in commands, **set** and **unset**, that are related in some sense but in another sense are just confusing.

The easy one is **unset**. [Example 11-2](ch11.html#ch11ex02) shows how to use this to destroy a variable.

**Example 11-2** Using **unset** to Destroy a Variable

$ FOO=bar  
$ echo $FOO  
bar  
$ unset FOO  
$ echo $FOO

Note that when you’re unsetting, refer to the variable name without the dollar sign.

**set** is a multitalented command. By itself it shows all variables and functions in the environment. Contrast this to **export**, which only shows exported variables.

The next use of **set** is to enable or disable features in the shell. For example, **set -x** tells the shell to print each command as it is executed. At the shell this isn’t that useful, but you can use this in a shell script for debugging. To revert the behavior, use **set +x**. The alternate way to set this is with **set -o xtrace**.

A useful option is called **noclobber**, which is enabled with **set -C** or **set -o noclobber**, as shown in [Example 11-3](ch11.html#ch11ex03). This tells the shell not to overwrite

existing files with redirects.

**Example 11-3** Using **noclobber**

[**Click here to view code image**](ch11_images.html#p11ex03a)

$ echo hi > test  
$ echo hi > test  
$ set -o noclobber  
$ echo hi > test  
-bash: test: cannot overwrite existing file

The first command in the preceding example puts the string **hi** into a file named **test**. It’s done once more to show that the shell will let you overwrite a file. The **noclobber** option is enabled and the same test is run, which results in an error. To disable **noclobber** run **set +o noclobber**.

The final use of **set** is to assign [*positional parameters*](gloss01.html#gloss_308). Within the shell are reserved variable names. Among those are **$1** and other numbered variables, which are used to pass information to scripts from the command line and other functions. Anything not passed to **set** and recognized as an option is assigned to one of these positional parameters:

[**Click here to view code image**](ch11_images.html#p318pro01a)

$ **set a b c d e f g h i j k l m**  
$ **echo $1**  
a  
$ **echo ${10}**  
j

Note the curly braces when trying to expand **$10**. Without them the shell thinks you’re trying to expand $1 and then appending a zero, so you would end up with **a0**.